NMPC Application using JModelica.org: Features and Performance

Christian Hartlep[1] Toivo Henningsson[2]

[1]Siemens AG, Germany, christian.hartlep.ext@siemens.com
[2]Modelon AB, Sweden, toivo.henningsson@modelon.com

Abstract

In the past JModelica.org was successfully applied for generating optimal trajectories. Using it for Nonlinear Model Predictive Control (NMPC) is the natural next step and sets high requirements on calculation time. To improve real time capabilities warmstarting of the optimization and elimination of algebraic variables based on Block Lower Triangular (BLT) form were implemented. In performance comparisons, using the example of steam temperature control, a speed-up of the optimization time by a factor of five and of two respectively was measured. The increased efficiency allows application of NMPC to faster systems than before.

Keywords: NMPC, BLT, IPOPT, JModelica.org

1 Introduction

A complete computational framework for Nonlinear Model Predictive Control (NMPC) demands various features provided by the FMI standard and its tool implementation, here JModelica.org by Modelon AB. This includes FMI2.0 linearization and elimination of algebraic variables based on the Block Lower Triangular (BLT) form, which recently became available. In the field of optimization it is intended to apply Modelica tools to large systems with safety and real-time restrictions. Effective equation pre-processing and efficient solving is demanded. For the latter warmstarting of optimizations improved real-time capabilities significantly. In this article the performance impact of different features is analyzed using the example of steam temperature control.

In Section 2 the NMPC framework is described with a focus on how JModelica.org tools are used for it. Recent enhancements for this particular application are discussed theoretically in Section 3 and analyzed regarding performance in Section 4. Finally Section 5 gives a short summary.

2 NMPC Loop Implementation

JModelica.org offers a basic framework for NMPC, which is described in (Axelsson et al., 2015). Here it is not used due to specific requirements for the intended application. This section gives an overview of this NMPC framework and how JModelica.org tools are used in it.

2.1 General Framework

The NMPC framework consists mainly of an observer and an optimizer, which interact with the plant as shown in Figure 1. In the top part the observer is depicted, which simulates the observer model based on the corrected state $\hat{x}^+_{k+1}$ of the previous time step and the current boundary conditions p_k. It generates a predicted state $\hat{x}^-_{k+1}$ and predicted plant output $\hat{y}_{k+1}$, which is given to the corrector. For Kalman filter based observers and Luenberger observer the filter gain calculation requires linearization. In dependence of the difference between predicted and actual plant output y^{Plant}_k the corrected state $\hat{x}^+_{k+1}$ is generated and used as initial state for the NMPC optimization. The NMPC generates the discrete future control input trajectory $u_{k,traj}$. In a post-processing step a simulation is performed to obtain continuous time control inputs $u^{NMPC}(t)$ for the plant. This allows excluding components which are relevant for the plant internal controllers but not the optimization. For this paper the plant is replaced by a simulation of a more complex model compared to the optimization model.

2.2 Observer

It is the observer's task to provide state estimates based on the estimated state of the previous time step and the plant outputs. Extended Kalman Filter (EKF) and Luenberger observer require the linearized state space representation for the calculation of the filter gain and a simulation of the nonlinear system. An Unscented Kalman Filter requires only the latter. The specific implementation is described in detail in (Bonvini et al., 2012). Implementing the EKF and Luenberger became more efficient recently due to the broad availability of FMI2.0 with

DOI
10.3384/ecp15118321

its ability to provide directional derivatives. Compared to the previous JModelica.org Model Unit (JMU)-based implementation the observer could be implemented more efficiently because simulation and linearization are handled by the same object. Benefits are reduced memory consumption and no need for transferring the predicted system state between the two objects.

2.3 Optimizer

Finding the optimal control inputs is handled by the JModelica.org optimization tool chain, which interfaces CasADi (Andersson, 2013) and IPOPT (Wächter and Biegler, 2006). It imports optimization problems written in Modelica/Optimica and discretizes them from continuous time to discrete time using collocation (Magnusson and Åkesson, 2012). Automatic generation of Jacobian and Hessian of the optimization problem is included. Recently, elimination of algebraic variables based on the BLT form of the resulting optimization problem was implemented to improve optimization solution time. Furthermore time-critical reoptimizations became faster due to reusage of the collocation and solver object and more importantly warmstarting of the optimization by providing primal and dual variables of the previous NMPC iteration. Both features are thoroughly explained in the next section. Another beneficial optimization feature is custom distribution of mesh elements which allows appropriate sizing of the optimization problem. At the beginning of the time horizon the element size is mainly determined by the required update interval, whereas at the horizon end only numerical accuracy remains important. An example of mesh relocation is given in (Zhao and Tsiotras, 2011).

3 JModelica.org Enhancements to Better Support NMPC Applications

This section describes some recent enhancements to the optimization tool chain in JModelica.org that are useful to improve convergence speed and robustness in the context of NMPC.

3.1 Warm Starting

The aim of warm starting is to reduce solution time and improve convergence robustness for repeated solution of similar optimization problems, such as the ones found in e.g. NMPC. In JModelica.org, these effects are achieved by two means: reusing the discretization and reusing the latest solution to improve the initial guess. To explain what the former means, we first give a short background on the discretization scheme, collocation.

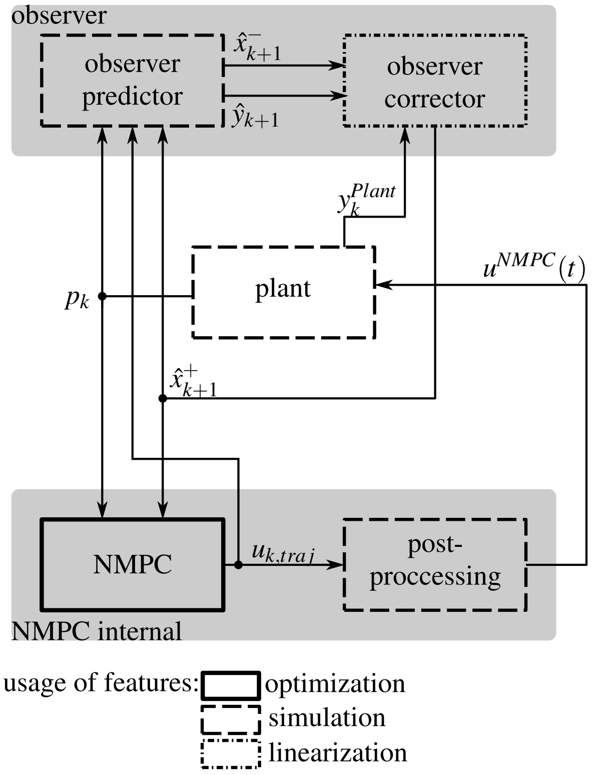

Figure 1. Structure of NMPC loop with signal exchanges and particular usage of JModelica.org feature

3.1.1 Collocation

When formulating an NMPC problem in Modelica/Optimica, modelling is naturally done in continuous time. To solve an optimization problem, we first *discretize* it by approximating it by a (large but sparse) Nonlinear Program (NLP) using collocation.

The time horizon is partitioned into *elements* (time intervals), and each time varying variable is approximated by a low order polynomial over each element. Each polynomial piece is described by sample values at a number of *collocation points* within the element.

3.1.2 Reusing the discretization

Creating the discretization can take a significant part of total solution time, sometimes even dominating it. To enable reuse, it must be valid not only for a single problem instance, but be parametrized by the degrees of freedom that can change from one time step to the next.

NLP parameters are introduced for

- Initial states.
- Parameters in the original model.
- External signals fed into the model, which can be used e.g. for time varying reference signals, or measurement signals in Moving Horizon Estimation.
- Scaling factors for equations.

The first three can be changed by the user between successive optimizations, while the last is typically determined automatically at the start of each optimization.

3.1.3 Reusing the latest solution

In NMPC, it is not only the optimization problems that are similar between time steps, but usually also their solutions. This feature can be exploited to minimize the amount of work for the nonlinear solver to find the solution to the next problem. We do this by

- Reusing primal and dual variables from the last solution to create the initial guess for the next. The primal variables represent the actual solution to the optimization problem, i.e. the trajectories of all variables, and can be shifted in time. If the last optimization failed, the solution from the last successful optimization can be used instead.

  The dual variables represent the cost of violating the constraints that are active in the optimization problem (see e.g. (Boyd and Vandenberghe, 2004)). Primal-dual optimization algorithms such as IPOPT converge both the primal and dual variables simultaneously, so fast convergence requires a good initial guess for the dual variables as well. Since it is not obvious that shifting is applicable to dual variables, they are reused as is.

- Reusing the nonlinear solver state itself, which may contain factorizations etc. that might be reused from one time step to the next.

For IPOPT, which is an interior point solver, we also need to adjust the initial value of the barrier parameter μ so as not to push the initial guess too far away from the constraints.

3.2 Elimination of Algebraic Variables

Modelica models often contain very many algebraic variables. In simulation, values for all algebraic variables must be explicitly solved at each time step. In optimization based on collocation, on the other hand, it possible to leave algebraic variables and the constraints that define them in the NLP, which will also contain constraints for the dynamics of the states.

Still, one can usually solve for some algebraic variables explicitly in terms of other, non-eliminated variables. Doing so brings a number of potential benefits for solving the NLP:

- Fewer decision variables to converge, to provide scalings and initial guesses for.

- Smaller matrices to factorize; factorization typically dominates the NLP solution time.

- Reduced sensitivity to the detailed formulation of the original Modelica model. Many formulations will yield equal NLPs, since intermediate variables introduced by the modeler can often be eliminated, producing the same results as if the intermediate was manually replaced by its definition.

Some variables might be left better uneliminated, however, if they can only be solved for iteratively or the elimination reduces sparsity too much in the NLP.

When importing a model for optimization in JModelica.org, an option has been introduced to eliminate algebraic variables. Suitable variables and equations are identified from the Block Lower Triangular (BLT) form of the model (the BLT form is constructed using Tarjan's Algorithm (Duff and Reid, 1978), starting from a perfect matching of the algebraic equation system).

The BLT form consists of an ordering of the equations and algebraic variables (non-states) in the algebraic equation system that must be solved to carry the solution of the system dynamics forward. It also contains a grouping of these equations and variables into diagonal blocks. To solve the entire equation system, each diagonal block can be solved in sequence, solving the system comprised of the block's equations for its unknowns. If a block requires the value of a variable outside of the block to solve it, that variable will already have been solved for in a previous block.

It is quite common for an algebraic variable to be expressed as an explicit function of one or more states and algebraic variables that can be computed before it. The relation between the algebraic variable and other variables will then be expressed by a scalar (1×1) BLT block with a single equation that the compiler can solve analytically; this is currently the only case used for elimination of algebraic variables. Elimination means that a variable will be calculated on the fly whenever needed, instead of requiring a nonlinear equation solver to iterate to find a solution for it. Variables with bounds are not eliminated since this will not reduce the size of the NLP.

As a simple example, consider the dynamic optimization problem

$$\begin{aligned} &\min_{c,x,u} \int_{t=0}^{1} c\,dt \\ &\text{s.t. } c = x^2 + u^2, \quad \dot{x} = u, \quad x(0) = 1 \end{aligned}$$

where c, x, and u are to be determined as real valued functions of time. The algebraic equation system is comprised of the two constraints $c = x^2 + u^2$ and $\dot{x} = u$, with unknowns c and $\dot{x}$. The state derivative $\dot{x}$ can only be solved from the latter equation, which leaves c to be solved from the former. This gives a BLT form with two scalar blocks.

The instantaneous cost c can be seen as an intermediate variable, and the BLT correctly identifies the opportunity to eliminate it, by matching it with the scalar equation $c = x^2 + u^2$. Using this equation to explicitly solve for and eliminate the algebraic variable c results in the optimization problem

$$\begin{aligned} &\min_{x,u} \int_{t=0}^{1} x^2 + u^2 \, dt \\ &\text{s.t. } \dot{x} = u, \quad x(0) = 1 \end{aligned}$$

which is smaller, with fewer equality constraints to satisfy and variables to converge.

In this case, the elimination of c yields additional benefits. The cost function in an optimization problem should be convex in a vicinity of the optimum (in this case, $x^2 + u^2$ is convex everywhere). Eliminating the intermediate variable c exposes this structure to the optimizer. In contrast, the original constraint $c = x^2 + u^2$ is non-convex even at the optimum, so keeping it will likely make it harder for the optimizer to find its way.

4 Performance Comparisons for Steam Temperature Control

In this section the performance impact of the discussed features is analyzed for the steam temperature control in a combined cycle power plant.

4.1 Test Setup

This section gives an overview of the optimization and plant models as well as the chosen scenario.

4.1.1 Soft- and Hardware

As software tools JModelica.org 1.16 (Magnusson and Åkesson, 2015), CasADi 1.9 with optimizer IPOPT 3.12 and HSL MA27 (HSL, 2013) as linear solver were used. The tests were carried out on a Intel i7 2620M (2.7 GHz) processor with Windows 7 x86.

4.1.2 Optimization Model

Figure 2 shows the optimization model with its three superheaters, four water injectors and two temperature sensors. Heat is transfered from flue gas (solid line) to two different steam lines (dash-dotted lines). The top and bottom superheater are part of the high pressure line (thickest dash-dotted line) and the middle one is part of the reheat line (thick dash-dotted line). It is the control target to provide a required steam temperature at the temperature sensors with minimal entropy production by changing the water valve openings. The optimization problem becomes nonlinear due to nonlinear water steam tables, nonlinear heat transfer functions in superheaters and the nonlinear objective function. In total the model includes six numerical states representing the physics, four control inputs and two outputs.

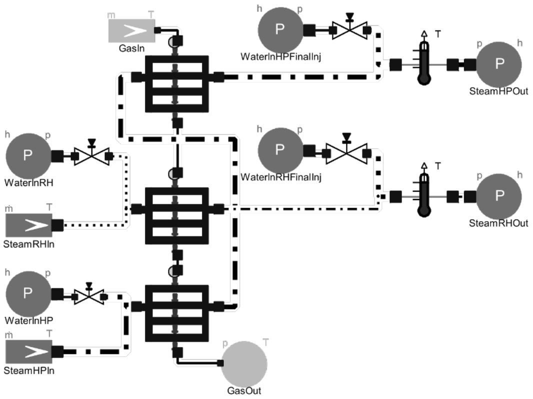

Figure 2. Optimization model for steam temperature control with superheaters, valves and temperature sensors

4.1.3 Plant Model

The performance of the NMPC loop is not tested on a real life plant but on a much more detailed model, which has been validated against measurement data of a real plant. This detailed plant model not only includes more heat exchangers than the model used for optimization (seven heat exchangers instead of three), but also intermediate piping between the heat exchangers as well as the underlying valve opening controllers for the injection valves. This results in a system of equations having about 14000 equations and 700 numeric states.

4.1.4 Test Scenario

As test scenario a plant startup is used because it includes various changes of setpoints and boundary conditions, which are not known in advance for the optimizer. Since the plant state changes drastically during this phase prediction accuracy is impaired especially at the beginning. The scenario is stopped before the plant reaches full load to capture only the dynamic behaviour and hence more challenging phases. Otherwise the results would be skewed in favour of the warmstart option.

In Figure 3 the resulting trajectories are plotted. The first two plots show measured, estimated and setpoint temperature for high pressure and reheat steam line respectively. The bottom subplot depicts the valve openings for intermediate (Int) and final injectors for each steam line. For the reheat part oscillations are visible, introduced by the stiff filter tuning. Such a filter tuning should not be used for a real plant but is beneficial for the performance test due to the stronger initial state update, which acts as disturbance for the optimizer.

4.2 Performance Comparison

For the application of NMPC low calculation time is required. Therefore features described in Section 3 are

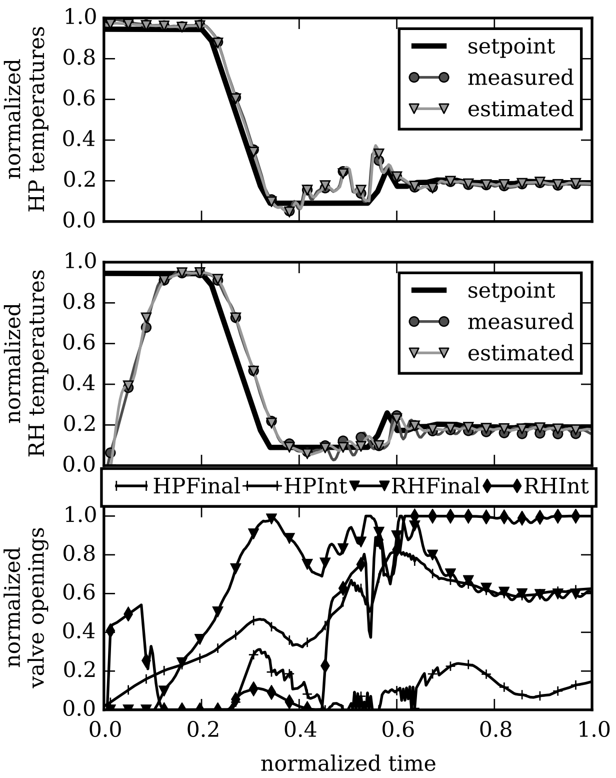

Figure 3. Test scenario for reheat (RH) and high pressure (HP) heat exchanger temperature control

tested here.

4.2.1 Criteria

Optimization performance is compared with regards to maximum and mean overall optimization time t_{opt}, maximum and mean IPOPT solution time t_{sol}, mean iterations n_{iter}, mean time per iteration. The overall optimization time includes setting initial trajectory and boundaries, JModelica.org preprocessing, optimization and JModelica.org post-processing. The gathered performance data is collected in Table 1.

4.2.2 Warmstart

Enabling the warmstart option improves performance in almost all performance criteria (see line 1 and 2 in Table 1) without changing the calculated trajectories. Warmstarting IPOPT reduced the average number of iterations and hence IPOPT solution time showed a reduction of 44%. Most notably is the reduction of the mean overall optimization time by 80% due to reusing of the solver object. An overview of the overall optimization time distribution is given in Figure 4. The distribution for warmstart is more narrow, which is visible in a reduction of the standard deviation from 0.4 s for the normal start to 0.25 s. Thus the optimization time became more predictable. Also the number of outliers is lower in the warmstart distribution.

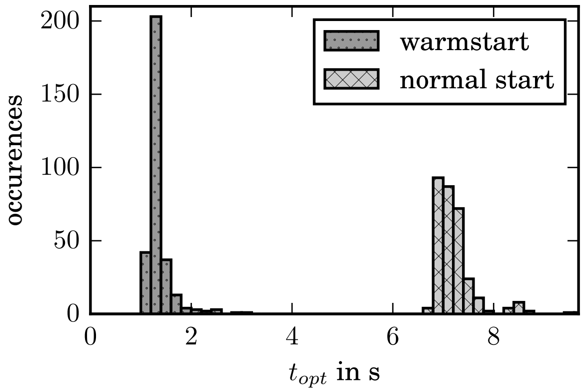

Figure 4. Comparison of total optimization time between normal optimization start and warmstart

4.2.3 BLT Elimination

Using BLT elimination improved performance and robustness in this case (see line 2 and 3 in Table 1). The most visible effect is the reduction of the number of variables by a factor of two (from 10359 to 5263, as reported by IPOPT). The smaller problem dimension translates into faster solution time and overall optimization time by about the same factor. The expected improvement in robustness was observed since 16 of 310 optimization were unsuccessful with disabled BLT, whereas all optimizations were successful with enabled BLT. Higher time per iteration shows that the original problem formulation was suboptimal.

4.2.4 Scaling of Computational Cost

Scaling of computational cost is compared for different scopes of included components. The smaller comparison model includes only the last high pressure superheater of the high pressure line (the top one in Figure 2) and water injectors before and after. This system has two numerical states, two control inputs and one control output.

Number of variables in IPOPT is 3.6 times higher for the model with both steam lines compared to the version with only one steam line. For the tests BLT and warmstart are enabled and the results are summarized in line 4 of Table 1. The average overall optimization, average solution time and average time per iteration are 2.9, 3.6 and 2.4 times higher respectively for the larger model. These values are slightly lower than the increase in optimization problem size which means computational cost increases less than linear.

4.2.5 Code Analysis

In the last part of the analysis distribution of computation time to different subtasks, which are performed in

	Modelled Steam Lines	BLT	warmstart	$\max(t_{opt})$	$\bar{t}_{opt}$	$\max(t_{sol})$	$\bar{t}_{sol}$	$\bar{n}_{iter}$	$\bar{n}_{iter}/\bar{t}_{sol}$
1	Reheat and High Pressure	on	off	8.06 s	7.00 s	1.04 s	0.81 s	20.76	38.9 ms
2	Reheat and High Pressure	on	on	3.03 s	1.37 s	2.09 s	0.45 s	10.37	43.5 ms
3	Reheat and High Pressure	off	on	15.14 s	2.78 s	13.98 s	1.00 s	17.64	56.6 ms
4	only High Pressure	on	on	1.06 s	0.48 s	0.59 s	0.13 s	7.21	18.0 ms

Table 1. Performance comparison of different optimization setups

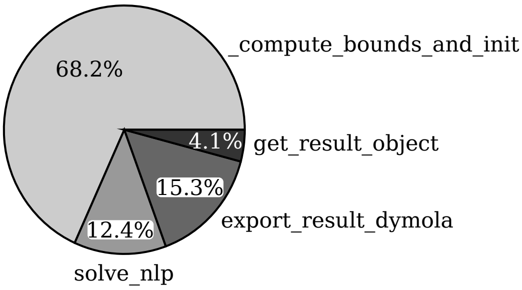

Figure 5. Percentage-wise breakdown of overall optimization time to different subtasks

every NMPC loop iteration, is analyzed. The result is visualized in Figure 5 for the computationally most expensive subtasks. All remaining subtasks take less than 2% together of the overall optimization time. Warmstart and BLT are enabled. About 32% of the overall optimization time is spent inside the optimize command of JModelica.org. Hence significant improvements in real time capabilities are possible going forward. Most time is spent for setting the initial trajectory (_compute_bounds_and_init). This step is not necessary for every NMPC loop iteration, but in case of an unsuccessful last optimization it is better to use the values of the last converged solution instead of the values that are still stored from the previous optimization. Therefore it is relevant for the analysis. Another time demanding task is exporting the result to a text file readable by Dymola. Whereas it is helpful for debugging the optimization result it could be omitted for time critical operations.

5 Summary

In conclusion the newly implemented features warmstart and BLT elimination in JModelica.org increased performance considerably for NMPC applications. JModelica.org with its Python interface proved to be viable choice for implementing an NMPC loop and was successfully tested for steam temperature control. Especially the warmstarting of the optimization improved tool capabilities in the field of NMPC application due to smaller optimization overhead and faster convergence. Variable elimination based on BLT reduced the optimization problem size and also gave a robustness improvement in the tests.

6 Acknowledgement

We would like to thank German Ministry BMBF for partially funding this work within the ITEA2 project MODRIO, as well as Siemens AG for providing further resources. For their critical responses and help we thank Prof. Pu Li, Kilian Link, Stephanie Gallardo Yances and Karin Dietl.

References

HSL, a collection of fortran codes for large-scale scientific computation, 2013. URL http://www.hsl.rl.ac.uk/.

Joel Andersson. *A General-Purpose Software Framework for Dynamic Optimization*. PhD thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium, October 2013.

Magdalena Axelsson, Frederik Magnusson, and Toivo Henningsson. A framework for nonlinear model predictive control in jmodelica.org. *Proceedings of the 11th International ModelicaConference*, 2015.

Marco Bonvini, Michael Wetter, and Michael D Sohn. An fmi-based framework for state and parameter estimation. *Proceedings of the 10th International ModelicaConference*, 2012.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521833787.

I. S. Duff and J. K. Reid. An implementation of tarjan's algorithm for the block triangularization of a matrix. *ACM Trans. Math. Softw.*, 4(2):137–147, June 1978. ISSN 0098-3500. doi:10.1145/355780.355785. URL http://doi.acm.org/10.1145/355780.355785.

Fredrik Magnusson and Johan Åkesson. Collocation methods for optimization in a modelica environment. In *Proceedings of the 9th International Modelica Conference*, 2012.

Fredrik Magnusson and Johan Åkesson. Dynamic optimization in jmodelica.org. *Processes*, 3(2):471, 2015. ISSN 2227-9717. doi:10.3390/pr3020471. URL http://www.mdpi.com/2227-9717/3/2/471.

Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006. ISSN 0025-5610. doi:10.1007/s10107-004-0559-y. URL `http://dx.doi.org/10.1007/s10107-004-0559-y`.

Yiming Zhao and Panagiotis Tsiotras. Density functions for mesh refinement in numerical optimal control. *Journal of guidance, control, and dynamics*, 34(1):271–277, 2011.